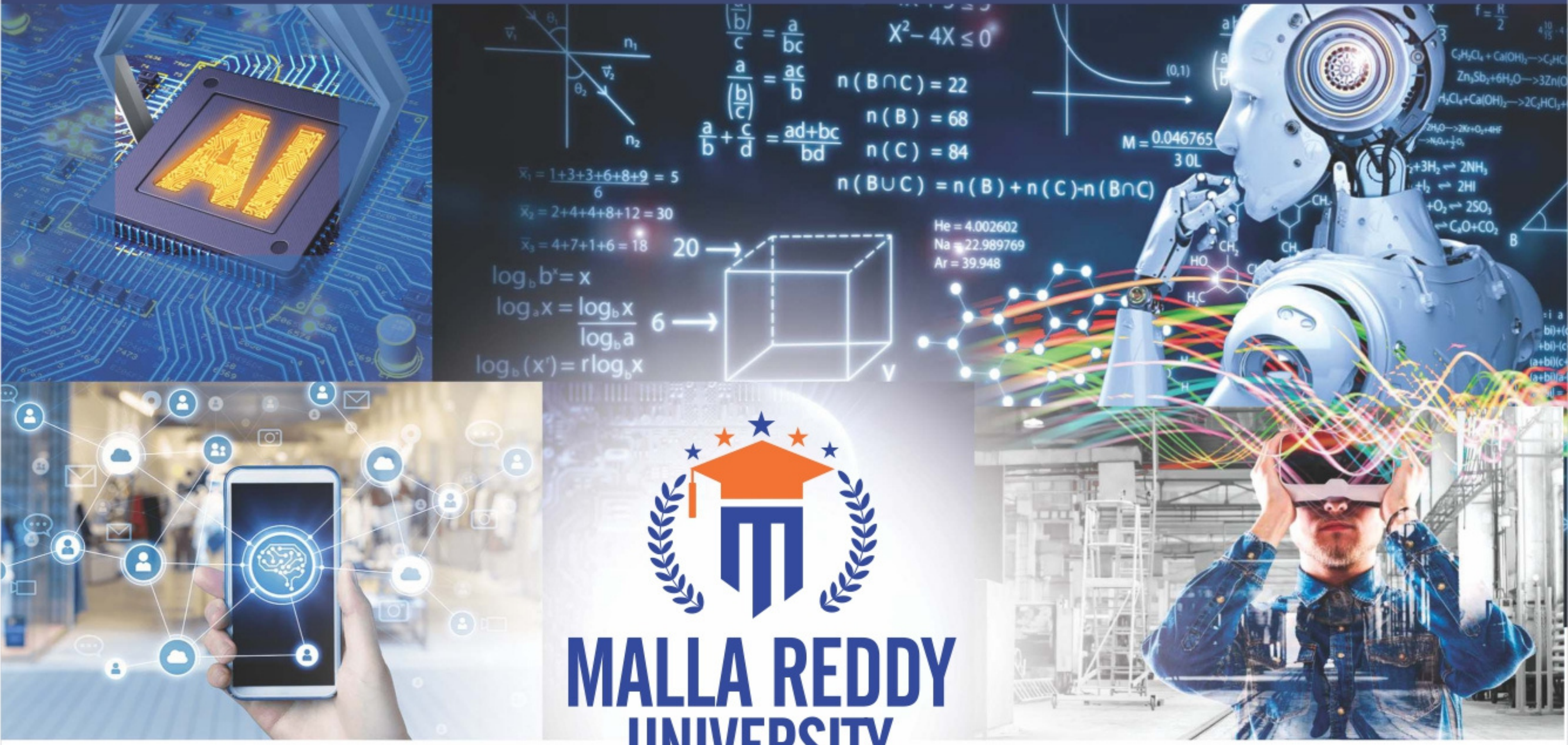
KNOWLEDGE EMPOWERMENT WORLD



MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and

Mulesoft Development Manual

B.Tech: IV Year I Semester

(CSE-AI&ML)

(2025-26)

School of Engineering

www.mallareddyuniversity.ac.in

Learning Manual

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Name			
Roll No.		Branch	
Year		Sem.	

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Maisammaguda, Kompally,
Medchal - Malkajgiri District
Hyderabad - 500100, Telangana State.
mruh@mallareddyuniversity.ac.in
www.mallareddyuniversity.ac.in

Certificate

School of Engineering

Certified that this is the bonafide record of practical work done by

Mr./Ms.. Roll. No........................ of

B.Tech .... year .... Semester for Academic year 20..... - 20..... in

.... Laboratory.

Date:

Faculty Incharge

CSE-AI & ML DEPARTMENT (IV YEAR I SEMESTER)

MULESOFT DEVELOPMENT MANUAL

INDEX

CSE-AI & ML DEPARTMENT (IV YEAR I SEMESTER)

MULESOFT DEVELOPMENT MANUAL

CONTENT

S.No	Name of the Experiment	Signature
1	**Experiment 1: Introduction to Anypoint Platform and Application Networks**	
2	**Experiment 2: API Design Using RAML in Anypoint Platform**	
3	**Experiment 3: Configuring Mule Applications and Accessing Databases Using Connectors**	
4	**Experiment 4: API Deployment and Management with API Manager**	
5	**Experiment 5: Logging and Debugging Mule Events**	
6	**Experiment 6: Mule Application Structure and Property Configuration**	
7	**Experiment 7: Consuming REST and SOAP Web Services**	
8	**Experiment 8: Event Flow Control and Routing**	
9	**Experiment 9: Error Handling and Reconnection Strategies**	
10	**Experiment 10: File Handling and Batch Processing Uing Flow Triggers**	

WEEK 1

Experiment 1: Introduction to Anypoint Platform and Application Networks

Objective: To explore the Anypoint Platform and understand the concept of Application Networks and API-Led Connectivity.

Theory: Anypoint Platform by MuleSoft is a unified integration platform for connecting applications, data, and devices. Application Networks are structured ways of linking applications via APIs using reusable and discoverable components. API-Led Connectivity uses three layers (Experience, Process, and System APIs) to ensure modular, scalable integration.

Step-by-Step Configuration:

1. **Login to Anypoint Platform**
    - Go to https://anypoint.mulesoft.com
    - Use your credentials to log in.
2. **Navigate to Design Center**
    - Click on "Design Center"
    - Select "Create" and choose "New API Specification"
    - Choose RAML and give a meaningful project name
3. **Create a Basic RAML API**

```
#%RAML 1.0
title: User API
version: v1
baseUri: http://localhost:8081/api
/users:
  get:
```

```
responses:
  200:
    body:
      application/json:
        example: |
[ { "id": 1, "name": "John Doe" } ]
```

- Click **Save**

4. **Mock the API**
   - Click "Mock this API"
   - An endpoint URL will be generated
5. **Use API Console to Test**
   - Try making a GET request to /users
6. **Explore Exchange**
   - Click "Exchange" on the left panel
   - Search for APIs and connectors
7. **Publish to Exchange**
   - From the Design Center, click "Publish to Exchange"
   - Add name, description, and save
8. **Navigate to API Manager**
   - Select the published API
   - Click "Manage API"
   - Choose a runtime (e.g., Mule 4)
   - Configure policies (e.g., Rate Limiting)

9. **Secure API with Client ID Enforcement**
    - Add "Client ID Enforcement" from Policies
    - Save and deploy
10. **Test Secure API Access**

- Use Postman to add headers:
    - client_id: <your-client-id>
    - client_secret: <your-client-secret>

Result: Successfully explored Anypoint Platform and created a sample Application Network using API-Led Connectivity.

Exercises:

1. Log in to Anypoint Platform and create a new API specification using RAML.
2. Create a basic GET API named Student API with an endpoint /students.
3. Mock the API and test using API Console.
4. Publish the API to Exchange.
5. Add “Client ID Enforcement” policy and test access with Postman.

Viva Questions:

1. What is the purpose of API-Led Connectivity?
2. What are the benefits of using Anypoint Platform?
3. What are Experience, Process, and System APIs?

WEEK 2

Experiment 2: API Design Using RAML in Anypoint Platform

Objective: To define and design a RESTful API using RAML in Anypoint Platform and test it using a Mock API.

Theory: RAML (RESTful API Modeling Language) is a YAML-based language for describing REST APIs. RAML helps in designing APIs before implementation. It supports mocking, documentation, and automatic discovery within the Anypoint Exchange.

Step-by-Step Configuration:

1. **Login to Anypoint Platform**
   - Visit https://anypoint.mulesoft.com and log in.
2. **Go to Design Center**
   - Click "Design Center" > "Create" > "New API Specification"
   - Choose RAML 1.0, provide a name like product-api.
3. **Design RAML Structure**

```
#%RAML 1.0
title: Product API
version: v1
baseUri: http://localhost:8081/api
/products:
  get:
    description: Get all products
    responses:
      200:
        body:
          application/json:
            example: |
              [ { "id": 1, "name": "Laptop", "price": 50000 } ]
  post:
    description: Create a new product
    body:
      application/json:
        example: {
          "id": 2,
          "name": "Mobile",
          "price": 25000
        }
    responses:
      201:
        body:
          application/json:
            example: { "message": "Product created" }
```

- Save the API specification

4. **Test with Mocking Service**
   - Click "Mock this API"
   - Copy the mock URL provided
   - Test with Postman using GET and POST operations
5. **Publish to Exchange**
   - Click "Publish to Exchange"

- Add details like name, category, version

6. **Discover API in Exchange**
    - Open "Exchange"
    - Search and open the newly published API
7. **Create a Public Developer Portal**
    - In Exchange, select the API
    - Click on "Public Portal"
    - Customize and share with external developers
8. **Test Discoverability**
    - Use another account or incognito window to test API access

Result: Designed and mocked a RESTful API using RAML in Anypoint Design Center. Published the API and verified its availability in Exchange and the Developer Portal.

Exercises:

1. Design a RAML-based API named Library API with GET and POST for /books.
2. Provide examples for both methods and mock the API.
3. Publish the API to Exchange and create a Public Developer Portal.
4. Test discoverability from an incognito window.

Viva Questions:

1. What is RAML?
2. What is the advantage of API mocking?
3. What is the use of a Developer Portal?

WEEK 3

Experiment 3: Configuring Mule Applications and Accessing Databases Using Connectors

Objective: To configure a Mule application to connect with a database using Database Connector in Anypoint Studio and retrieve data using a RESTful API.

Theory: Mule Applications can interact with external databases using connectors. MuleSoft's Database Connector supports querying, inserting, updating, and deleting data from relational databases like MySQL, Oracle, and SQL Server. This experiment demonstrates how to build a REST API that fetches data from a MySQL database.

Prerequisites:

- Anypoint Studio Installed
- MySQL Server with test database
- JDBC Driver for MySQL

Step-by-Step Configuration:

1. **Create a New Mule Project**
    - Open Anypoint Studio
    - File > New > Mule Project
    - Project Name: db-access-api
2. **Add HTTP Listener**
    - Drag an HTTP Listener to the canvas from Mule Palette
    - Configure:
        - Path: /products
        - Host: 0.0.0.0
        - Port: 8081
3. **Add Database Connector**

- In Mule Palette, search and drag **Database > Select** next to the HTTP Listener
- Click on the connector and create a new **Global Element**:
    - Connection Type: MySQL Connection
    - Host: localhost
    - Port: 3306
    - User: root
    - Password: yourpassword
    - Database: productdb
    - Add JDBC Driver if prompted

4. **Configure SQL Query**

- In the SQL Query field, type:

```
SELECT * FROM products;
```

5. **Transform Data to JSON**

- Drag a **Transform Message** component
- Auto-generate DataWeave expression:

```
%dw 2.0

output application/json

---

payload
```

6. **Run the Application**

- Click Run > Run Project db-access-api
- Open Postman and send a GET request to:

http://localhost:8081/products

7. **Check Response**
    - The response will show data from your MySQL products table in JSON format
8. **Handle Errors (Optional)**
    - Add an **Error Handler** below the main flow
    - Configure a **Logger** to print custom error messages

Result: Configured a Mule application to access a MySQL database using the Database Connector and exposed the data through an HTTP API.

Exercises:

1. Create a Mule application to retrieve records from a students table.
2. Use MySQL as the database with appropriate credentials.
3. Use Transform Message to convert the result to JSON.
4. Test the API using Postman.

Viva Questions:

1. What are the steps to connect a database in Mule?
2. Which JDBC driver is required for MySQL?
3. How is a query result transformed into JSON in Mule?

WEEK 4

Experiment 4: API Deployment and Management with API Manager

Objective: To deploy a Mule API and manage it using API Manager on the Anypoint Platform by applying policies and securing access.

Theory: Anypoint API Manager allows you to manage APIs throughout their lifecycle, including deployment, policy enforcement, analytics, and security. It helps in applying rules such as rate limiting, IP filtering, and client enforcement without changing the API code.

Step-by-Step Configuration:

1. **Prepare a Mule API for Deployment**
    - Use Anypoint Studio to build a basic API
    - Example Endpoint: /greeting
    - Add HTTP Listener and Transform Message:
    - %dw 2.0
    - output application/json
    - ---

{ message: "Hello from deployed API!" }

2. **Deploy to CloudHub**
    - In Studio, right-click project > Anypoint Platform > Deploy to CloudHub
    - Provide domain name and runtime (e.g., Mule 4.4.0)
    - Set environment (Sandbox/Production)
    - Click Deploy
3. **Access Deployed API**
    - Once deployed, use the domain:

https://<your-app-name>.cloudhub.io/greeting

- Verify response in browser or Postman

4. **Register API in API Manager**
    - Go to Anypoint Platform > API Manager
    - Click "Manage API > Manage API from Exchange"
    - Choose the published API from Exchange
    - Select appropriate environment and version
5. **Apply API Policies**
    - Inside API Manager, click "Policies"
    - Add "Client ID Enforcement"
    - Set up required parameters
6. **Test Policy Enforcement**
    - Use Postman to send requests without client credentials (should fail)
    - Add headers:
        - client_id: your-client-id
        - client_secret: your-client-secret
    - Retest (should succeed)
7. **Analyze API Usage**
    - Go to "Analytics" tab in API Manager
    - View requests, errors, response times
8. **Add SLA Tiers (Optional)**
    - Inside the API Manager > SLA Tiers
    - Define plans for different usage levels (Free, Premium, etc.)
    - Approve/deny app access manually or automatically

Result: Successfully deployed a Mule API to CloudHub and managed it using API Manager with security policies.

Exercises:

1. Deploy an API (e.g., /hello) to CloudHub.
2. Access and test the deployed endpoint.
3. Register the API in API Manager and apply Rate Limiting and Client ID Enforcement.
4. Analyze usage in the Analytics tab

Viva Questions:

1. What is the role of API Manager?
2. How does policy enforcement work in Anypoint Platform?
3. What is the purpose of SLA tiers?

WEEK 5

Experiment 5: Logging and Debugging Mule Events

Objective: To understand and implement logging, debugging, and variable management in Mule applications using Anypoint Studio.

Theory: Mule Events consist of a message, attributes, and variables. Logging is used to trace the flow and data within an application. Debugging allows developers to inspect the behavior of applications and find issues during development. MuleSoft provides Logger, Set Variable, and Debugger for effective development and issue resolution.

Step-by-Step Configuration:

1. **Create a New Mule Project**
    - Open Anypoint Studio
    - File > New > Mule Project
    - Name: logging-debugging-app
2. **Add HTTP Listener**
    - Drag and drop an HTTP Listener
    - Configure path: /student
    - Port: 8081
3. **Add Set Payload**
    - Add a Set Payload component
    - Set output:

```
%dw 2.0
output application/json
---
{
  name: "Arun",
department: "CSE"
}
```

4. **Add Logger**
   - Add a Logger component after the Set Payload
   - Set message: Payload before transformation: #[payload]
   - Log level: INFO
5. **Add Transform Message**
   - Transform payload to add additional data:

```
%dw 2.0
output application/json
---
payload ++ { year: 2025 }
```

6. **Add Final Logger**
   - Add another Logger component after the transform
   - Log message: Transformed Payload: #[payload]
7. **Run the Application in Debug Mode**
   - Right-click on project > Debug As > Mule Application
   - Add breakpoints on components
   - Use Postman to send request: http://localhost:8081/student
   - Watch debugger step through each event
8. **Create and Use Variables**
   - Add Set Variable component before transform
   - Name: userType, Value: "admin"
   - Log variable: User type is #[vars.userType]

9. **Access Attributes**

   - Add logger to print request method:

HTTP Method: #[attributes.method]

10. **Use Mule Event Viewer**

    - Go to Debug Perspective
    - Open Event Viewer to track payload, variables, and attributes across the flow

Result: Successfully logged payloads, accessed attributes and variables, and debugged the Mule flow using Anypoint Studio tools.

Exercises:

1. Create a Mule flow with HTTP Listener (/employee) and Set Payload.
2. Add multiple Logger components before and after transformation.
3. Add a Set Variable and log its value.
4. Use Debug mode and track the execution via Event Viewer.

Viva Questions:

1. What are the key components of a Mule Event?
2. How do you log values in Mule applications?
3. What is the difference between payload, variables, and attributes?

WEEK 6

Experiment 6: Mule Application Structure and Property Configuration

Objective: To understand the structure of a Mule application project and configure properties for different environments.

Theory: A Mule application follows a standardized project structure. Key files include mule-artifact.json, pom.xml, and src/main/mule. Properties files allow the use of dynamic configuration values, making applications reusable across environments like Dev, Test, and Prod.

Step-by-Step Configuration:

1. **Create a New Mule Project**
    - Open Anypoint Studio
    - File > New > Mule Project
    - Project Name: property-config-app
2. **Explore Project Structure**
    - src/main/mule: Contains all Mule flows (XML files)
    - src/main/resources: Contains configuration files
    - pom.xml: Handles project dependencies
    - mule-artifact.json: Contains deployment metadata
3. **Create Property Files**
    - In src/main/resources, create:
        - config-dev.properties
        - config-prod.properties
    - Example content:

      app.message=Welcome to Development Environment

      app.message=Welcome to Production Environment

4. **Configure Global Property Placeholder**
    - Go to Global Elements > Create > Configuration Properties
    - Name: config
    - File: config-dev.properties
5. **Create Flow Using Property**
    - Add HTTP Listener (Path: /env, Port: 8081)
    - Add Set Payload with:

      ```
      %dw 2.0
      output text/plain
      ---
      p('app.message')
      ```
6. **Switch Between Environments**
    - Right-click on config global element > Edit
    - Change to config-prod.properties and redeploy
    - Test again at http://localhost:8081/env
7. **Use Mule Property Files in pom.xml (Optional)**
    - Configure Maven profiles for environment-based builds
8. **Secure Property Files**
    - Use secure property placeholder module to encrypt values
    - Encrypt sensitive values using MuleSoft secure property tool
9. **Test the Flow**
    - Run the application
    - Use Postman or browser to verify the dynamic message

Result: Successfully understood Mule project structure and implemented environment-based configuration using property files.

Exercises:

1. Create a Mule project and explore the directory structure.
2. Create config-dev.properties and config-prod.properties with custom messages.
3. Use a property in Set Payload and switch between environments.
4. Optional: Secure one of the properties using Mule Secure Property Tool.

Viva Questions:

1. What is the role of mule-artifact.json?
2. How can property files be used for environment-specific settings?
3. What are the folders in a Mule project and their purposes?

WEEK 7

Experiment 7: Consuming REST and SOAP Web Services

Objective: To consume REST and SOAP web services in a Mule application using HTTP Request and Web Service Consumer components.

Theory: Mule applications can act as clients to external REST or SOAP web services using HTTP connectors. REST services use HTTP methods like GET, POST, etc., while SOAP services rely on WSDL-based XML communication. Anypoint Studio provides components for consuming both.

Step-by-Step Configuration:

Part A: Consuming a RESTful Web Service

1. **Create New Mule Project**
    - Project Name: consume-rest-api
2. **Add HTTP Listener**
    - Path: /weather
    - Port: 8081
3. **Add HTTP Request Connector**
    - Drag and drop HTTP Request next to listener
    - Method: GET
    - URL: https://api.open-meteo.com/v1/forecast?latitude=35&longitude=139&hourly=temperature_2m
4. **Add Transform Message**
    - Convert response to JSON:

```
%dw 2.0
output application/json
---
```

payload

5. **Run and Test**
    - Deploy the app and access: http://localhost:8081/weather
    - Verify response from external API

Part B: Consuming a SOAP Web Service

1. **Create New Mule Project**
    - Project Name: consume-soap-api
2. **Download WSDL**
    - Example WSDL: http://www.dneonline.com/calculator.asmx?WSDL
3. **Add Web Service Consumer**
    - Configure using WSDL file
    - Choose operation: Add
    - Set input parameters for integers
4. **Create Flow**
    - HTTP Listener at /add
    - Use Set Payload to define input:

```
%dw 2.0
output application/java
---
{
  intA: 5,
  intB: 7
}
```

5. **Add Web Service Consumer After Payload**
   - Map inputs automatically
6. **Transform Response**
   - Use Transform Message:

     ```
     %dw 2.0

     output application/json

     ---

     payload
     ```
7. **Test Using Postman**
   - Send GET request to http://localhost:8081/add
   - Output: { "AddResult": 12 }

Result: Successfully consumed both REST and SOAP web services using HTTP Request and Web Service Consumer in Anypoint Studio.

Exercises:

1. Create a flow /weather to consume weather data using REST (open-meteo).
2. Create another flow /add to consume a SOAP Calculator API for addition.
3. Use Transform Message to format responses for both.
4. Test both APIs using Postman.

Viva Questions:

1. What is the difference between REST and SOAP?
2. How do you consume an external REST API in Mule?
3. What is a WSDL and why is it important for SOAP?

WEEK 8

Experiment 8: Event Flow Control and Routing

Objective: To implement flow control techniques such as multicast, choice routing, and validation in a Mule application.

Theory: MuleSoft provides flow control components such as **Choice**, **Scatter-Gather (Multicast)**, and **Validation** for flexible message routing. These help direct the Mule event based on conditions or business logic, improving flow execution and error handling.

Step-by-Step Configuration:

1. **Create a New Mule Project**
    - Project Name: event-routing-app
2. **Add HTTP Listener**
    - Path: /student
    - Port: 8081
3. **Use Choice Router**
    - Drag Choice component after the listener
    - Add conditions:
        - When #[payload.age > 18] → Logger: “Eligible for College”
        - Else → Logger: “Still in School”
4. **Add Set Payload Before Choice**

    Set payload as:

```
%dw 2.0
output application/json
---
{
  name: "Raj",
  age: 20
}
```

5. **Test the Flow**
    - Run the app and access: http://localhost:8081/student
    - Verify logger output based on condition
6. **Implement Multicast Using Scatter-Gather**
    - Create a new flow: /broadcast
    - Add Scatter-Gather component
    - Inside, add two flows:
        - Logger 1: “Send to Academic Department”
        - Logger 2: “Send to Admin Department”
7. **Run and Access**
    - Use: http://localhost:8081/broadcast
    - Check logs to confirm both routes are triggered
8. **Validate Input Data**
    - Add a new flow: /validate
    - Use HTTP Listener + Validation: Is Not Null component
    - Set payload as:

```
%dw 2.0
output application/json
---
{
  studentId: "S001"
}
```

9. **Add Logger After Validation**
    - Message: “Validation Passed”
10. **Error Handling (Optional)**

- Add error handler for validation failure
- Logger: “Validation Failed: #[error.description]”

Result: Successfully implemented conditional routing, multicast event processing, and validation logic in Mule using Choice, Scatter-Gather, and Validation components.

Exercises:

1. Implement a Choice router to log whether a student is eligible for college.
2. Use Scatter-Gather to send a message to two loggers simultaneously.
3. Use Validation to check for null studentId in a payload.
4. Handle validation errors using error handlers.

Viva Questions:

1. What is the difference between Choice and Scatter-Gather?
2. How does Mule validate payloads?
3. What happens when validation fails?

WEEK 9

Experiment 9: Error Handling and Reconnection Strategies

Objective: To implement error handling scopes and reconnection strategies for system errors in Mule applications.

Theory: MuleSoft provides robust error handling mechanisms to manage and recover from application and system-level errors. Scopes like On Error Continue and On Error Propagate define how errors are handled. Reconnection strategies help applications recover from transient failures, such as database or network outages.

Step-by-Step Configuration:

1. **Create New Mule Project**
   - Project Name: error-handling-app
2. **Add HTTP Listener**
   - Path: /divide
   - Port: 8081
3. **Add Set Payload to Simulate Error**
   - Payload script:

     ```
     %dw 2.0
     output application/json
     ---
     {
       numerator: 10,
       denominator: 0
     }
     ```
4. **Add Transform Message**
   - Perform division:

```
%dw 2.0
output application/json
---
{
  result: payload.numerator / payload.denominator
}
```

- This will cause a divide-by-zero error

5. **Add Error Handler**
    - Drag Error Handler to flow
    - Inside it, add On Error Propagate
    - Logger: “Error occurred: #[error.description]”
    - Set Payload: “A calculation error occurred”
6. **Run and Test Error Response**
    - Use Postman: http://localhost:8081/divide
    - Response should be: “A calculation error occurred”
7. **Implement Reconnection Strategy**
    - Create another flow for DB access
    - Use Database Connector with wrong credentials
    - In the connector config, enable Reconnection Strategy:
        - Reconnect: 3 times
        - Frequency: 2000ms
8. **Logger in Error Flow**
    - Log #[error.message] to confirm reconnection attempts
9. **Switch to On Error Continue (Optional)**

- Use instead of propagate
- Allow flow to continue after logging error

10. **Handle HTTP Errors**

- Simulate 404 by calling a missing resource
- Add global error handler with HTTP:NOT_FOUND

Result: Successfully implemented error scopes and configured reconnection strategies to make the Mule application resilient to runtime failures.

Exercises:

1. Simulate a divide-by-zero error in a flow /divide and handle it using On Error Propagate.
2. Implement reconnection strategy in a Database Connector with invalid credentials.
3. Log error messages in both error scopes.
4. Switch to On Error Continue and observe behavior change.

Viva Questions:

1. What is the difference between On Error Continue and On Error Propagate?
2. How are reconnection strategies configured in Mule?
3. What happens if an error is not handled in a Mule flow?

WEEK 10

Experiment 10: File Handling and Batch Processing Using Flow Triggers

Objective: To create Mule flows that trigger based on file events and implement batch processing to manipulate records.

Theory: Flow triggers in MuleSoft allow integration events to begin when a file is created or updated. The File Connector can monitor directories. Batch processing is used to handle large datasets, breaking them into individual records using scopes like For Each and Batch Job.

Step-by-Step Configuration:

Part A: File Trigger and Object Store

1. **Create New Mule Project**
    - Project Name: file-trigger-app
2. **Add File Listener (On New File)**
    - Drag File connector → On New File
    - Directory: src/main/resources/inbox
    - File Pattern: *.txt
3. **Add Logger**
    - Message: "File Received: #[attributes.name]"
4. **Write File Output**
    - Use File > Write component
    - Directory: src/main/resources/outbox
    - File Name: processed-#[attributes.name]
    - Content: #[payload]
5. **Use Object Store to Track File Names**
    - Use Object Store Connector to store key: #[attributes.name]

- Configure expiration if needed

6. **Run and Drop File**
    - Place a .txt file in inbox folder
    - Check outbox folder for processed file

Part B: Batch Processing of Records

7. **Create a CSV File in src/main/resources/data.csv**

```
id,name,score
1,Alice,85
2,Bob,90
3,Charlie,78
```

8. **Create Flow to Read File**
    - Use File > Read
    - Path: src/main/resources/data.csv
9. **Use Batch Job Scope**
    - Add a Batch Job scope after File Read
    - Inside, use a For Each scope to iterate lines
10. **Transform CSV Records**

- Use Transform Message to convert CSV lines to objects

```
%dw 2.0
output application/json
import * from dw::core::Strings
---
payload splitBy "\n" map ((line, index) ->
  if (index != 0)
```

```
    { id: line[0], name: line[1], score: line[2] }
   else null
   ) filter ((item) -> item != null)
```

11. **Add Logger in Batch Step**

- Log each student's score

12. **Apply Filter to Score > 80 (Optional)**

- Add Choice or DataWeave filter expression

Result: Successfully triggered flows using file creation events and processed multiple records using batch and loop constructs.

Exercises:

1. Configure a flow to listen for .txt files in an inbox folder and write to outbox.
2. Use Object Store to keep track of processed files.
3. Create a CSV file and use a Batch Job to process each line.
4. Filter and log student records with score > 80.

Viva Questions:

1. How do flow triggers work in Mule?
2. What is the purpose of the Batch Job scope?
3. How can Object Store help in flow executions?

" A thought beyond horizons of success committed for professional excellence "

Vision

To be a world class university visualizing a great future for the young aspirants, with innovative nature, research culture and ethical sensitivities to meet the global challenges improving the quality of human life.

Mission

To impart value based futuristic higher education moulding students into globally competent empowered youth, rich in culture and ethics along with professional expertise.

To promote innovation, entrepreneurship, research and development for the broad purpose of fulfilling societal goals such as societal welfare and benefit.

Quality Policy

To pursue continual improvement of teaching learning process of Undergraduate, Post Graduate programs and Research Programs vigorously.

To provide state of the art infrastructure and expertise to impart the quality education.

To groom the students to become intellectually creative and professionally competitive.

To explore the opportunities in the professional fields.

Academic Best Practices

Industry Oriented Curriculum
Technology Training and Certifications
Institution to Corporate Exposure
Interactive Learning
International Career Guidance
Choice Based Flexible Curriculum

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Maisammaguda, Kompally, Hyderabad - 500 100
Telangana State

www.mallareddyuniversity.ac.in